Remarks

The Office Action dated June 22, 2004 has been noted, and its contents carefully studied. In light of the foregoing amendments and the following arguments, reconsideration of the rejection under 35 U.S.C. §102 and/or §103 is courteously requested.

To facilitate the Examiners' reconsideration, the following discussion of the invention is presented herein. More specifically, notwithstanding the Examiner maintaining the same rejections as before the Request for Continued Examination, it is not understood how the Examiner can be interpreting the cited references in the manner applied since the sections by column and lines cited to support the Office Action, upon detailed examination do not teach or even use the language alleged in the Office Action. Accordingly, it is courteously urged that this explanation is presented herein to educate the Examiner to attempt to eliminate any confusion the Examiner may have about the nature of the invention, particularly as it relates to the references cited.

Initially, the prior art is discussed as follows. As may be appreciated, current directory structure design places directory files in a compact, linear list within a directory data structure. This is what is done in the Johnson reference. When the file system accesses the directory structure, the file system reads the directory structure from the hard disk storage device and places the directory structure in memory buffer. The representation of the memory is a replica of the hard disk representation. When the file system requires a specific file, the file system performs a sequential search down the list of directory files. The file system inspects one filename at a time until the file is found or until the list is exhausted. In this regard, this is exactly what is described in column 8, lines 9-22 of the cited Johnson patent.

In contrast, the file access system and method of the invention allows multiple network computers to quickly search and retrieve requested files. The file access system relies on the existence of i-nodes, which are well known. The file access system adds a field to the i-node corresponding to directory cache hash table. The field contains a pointer. The file access system allocates memory for directory cache, and the directory cache contains a hash table. The pointer points to the hash table. The hash table contains an array of hash buckets. The hash table contains all of the files residing in a directory. The file i-node is hashed to a specific bucket in

the array of hash buckets. The specific bucket points to a list of files that may correspond to the requested file i-node. Each entry in the list of files contains a link to a corresponding offset where filenames are stored in memory. If the offset contains a matching filename, the requested file is found and the search is complete. If the offset does not contain a matching file name, then the file access system returns to the list of files and the next entry is checked. The method repeats until the filename is found. If the filename is not found, then the file name does not exist in the directory cache.

In this context, all of the claims have been amended to generally include these features, and are believed self-evident from a reading thereof. Thus, they need not be discussed in greater specific detail herein. Support for the amendments to the claims is found in the specification at paragraph 22 thereof.

Having thus generally discussed the invention, it is respectfully urged that the invention as now recited in the claims is not anticipated under 35 U.S.C. §102 or obvious under 35 U.S.C. §103 from the cited references, as is evident from the following detailed discussion of these references presented herein for the Examiner's kind consideration.

U.S. Patent No. 5,151,989 to Johnson et al.

U.S. Patent No. 5,151,989 to Johnson et al. (hereinafter "Johnson") teaches a directory caching technique for a plurality of data processing systems which are connected together in a network. Johnson recognizes that accessing files across a network presents two competing problems. One problem involves the time required to transmit data across the network for successive reads and writes. The second problem is that if the file data is stored in a node (computer on a network) to reduce network traffic, the file integrity may be lost. This may be caused because if one of the several nodes (computers on the network) is also writing to the file, the other nodes (computers on the network) accessing the file may not be accessing the latest updated file that has just been written (column 3, lines 42-49). The system described in Johnson is part of an operating system which provides a solution to the problem of managing distributed information (column 3, lines 55-58).

What Johnson teaches is a directory caching technique for a plurality of data processing systems which are connected together in a network. In the system of Johnson, when a local, or

client, data processing system interrogates a remote, or server, data processing system for a unit of directory information, the server is enabled to automatically send additional units of pertinent directory information back to the client system in response to a subsequent change in the directory structure of the server system. If the server system is unable to continue updating the client system, the server informs the client system of this fact, and the client system purges itself of the formerly stored directory cache entry relative to the particular path since the client system can no longer consider the cache path information to the be correct (column 5, lines 52-68). The essential aspects of the file system of Johnson are that each file on an individual file system is identified by its i-node number (a conventional implementation), and directories and files, and thus a directory can be identified by its i-node number. Thus, implementing a search following paths, Johnson merely teaches how operating systems use mounts of entire file systems to create file trees and how paths are followed in such a file tree (column 8, lines 5-8).

In Johnson, it is contemplated that when a client and server communicate about a server file, they need a way to identify the file. Files are identified by a file handle which contains the device number, i-node number, and i-node generation number. The reason for a file handle is that if a client makes a request to the server and gets a file handle in reply, the client stores and remembers the file handle. Some activity at the server causes the file to be deleted and the i-node slot becomes reused for another file. The client makes a request to the server using the stored file handle. The server receives the file handle and performs the operation on the new file. This would be an unacceptable operation (column 13, lines 52-68).

In accordance with the solution presented by Johnson, the file is represented by use of i-node generation number. The i-node generation number is stored on disk as a field in the i-node. When the server deletes a file it increments the i-node generation number. If a request arrives at a server, the file handle is broken apart, the device number and i-node are used to locate the i-node, and then the file i-node generation number is compared to the i-node's i-node generation number. If they are different, then the request is rejected.

As may be appreciated, this has nothing to do with the concept of every directory having an associated directory i-node memory structure with a field added to the i-node which corresponds to a directory cache hash table. In Applicants' invention the field itself contains a

pointer and the file access system allocates memory for directory cache with the directory cache containing a hash table. The pointer points to the hash table and the hash table contains an array of hash buckets. The hash table contains all of the files residing in the directory and the file i-node is hashed to a specific bucket in the array of hash buckets. The specific bucket points to a list of files and they correspond to the requested file i-node. Each entry in the list of files contains a link to a corresponding offset where file names are stored in memory. If the offset contains a matching filename, the requested file is found and the search is complete. If the offset does not contain a matching filename, then the file access system returns to the list of files and a next entry is checked. This method repeats until the file name is found. If the filename is not found, then the filename does not exist in the directory cache.

Johnson has nothing to do with the invention as claimed. Johnson merely addresses the issue of a server system being unable to update client systems and provides for a way of informing the client system of the fact that the client system cannot be updated for directories to be consistent with information or directories on the server system.

U.S. Patent No. 5,666,532 to Saks et al.

U.S. Patent No. 5,666,532 to Saks et al. (hereinafter "Saks") merely teaches a computer system having data organized in files, and having secondary storage for storing files, having a primary storage, and having one or more types of file subsystems for controlling transfer of files between primary storage and secondary storage. More specifically, Saks teaches a technique of writing to secondary storage using a delayed order write subsystem to control the order in which modifications are propagated to disks.

In terms of discussing buffering, there is nothing unique about what Saks teaches. More specifically, as with Johnson, Saks merely teaches conventional techniques wherein a process creates or opens a file by name. The UNIX file system provides that each component of the filename is parsed, and a check is made that the process has the necessary privileges for searching the intermediate directories. Thereafter, the i-node for the file is retrieved. I-nodes are stored in the file system and the UNIX file system reads the needed i-nodes into an i-node table that is maintained in primary storage for ease and efficiency of access (column 8, lines 14-28).

On the other hand, there is nothing in Saks which teaches or suggests a field in a directory i-node memory structure corresponding to a directory cache hash table with a pointer which points to the hash table. Yet still further, there is nothing in Saks wherein the hash table contains an array of hash buckets, and all the files residing in the directory such as that the file i-node is hashed to a specific bucket in the array. Yet still further, there is nothing in Saks which teaches that a specific bucket points to a list of files that may correspond to the requested file i-node with each entry in the list of files containing a link to a corresponding file offset where filenames are stored in memory. This allows for the fact that if the offset contains a matching filename, the requested file is found and the search is complete. On the other hand, if the offset does not contain a matching filename, then the file access system returns to the list of files and a next entry is checked, and the process is repeated until the filename is found. Ultimately, if the filename is not found, then the filename does not exist in the directory cache.

Again, Saks has nothing to do with these features of the invention including the offsets, the field added to the i-node and the combination with the pointers, hash tables and hash buckets as recited in Applicants' claims.

U.S. Patent No. 5,778,430 to Ish et al.

U.S. Patent No. 5,778,430 to Ish et al. (hereinafter "Ish") generally teaches a method and apparatus for determining whether a particular block which is the subject of a read request is present in cache at any particular time. The method employs a conventional hashing function which takes as its input a block number and outputs a hash index into a hash table with pointers. This is nothing more than the conventional hashing techniques employed in numerous computer systems, which is defined in the appended document. "Hashing" is the transformation of a string of characters into a usually short or fixed length value or key that represents the original string. Ish teaches conventional hashing and nothing more.

There is nothing in Ish which teaches a file access system which adds a field in a directory i-node structure corresponding to a directory cache hash table, with the field containing a pointer. Yet still further, there is nothing in Ish which teaches hashing a file i-node to a specific hash bucket in an array of hash buckets in a hash table. There is in fact no discussion of buckets in a hash table. Yet still further, there is no discussion or suggestion therein of having

the specific bucket point to a list of files that may correspond to the requested file i-node, with each entry in the list of files containing a link to a corresponding offset where filenames are stored in memory such that if the offset contains a matching filename, the requested file is found and the search is complete. Alternatively, in the invention if the offset does not contain a matching filename, the file access system returns to the list of files and a next entry is checked such that the method repeats until the filename is found. If the filename is not found, then the file name does not exist in the directory cache.

Accordingly, it is respectfully urged that the invention as recited in the amended claims is not anticipated or obvious from the references standing alone or in combination as proposed by the Examiner.

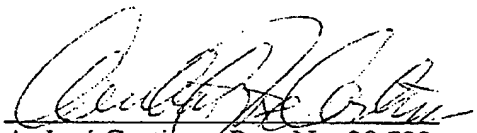
In fact, it is again respectfully urged that what the Examiner has done in establishing the rejection is a selective interpretation of the references in a hindsight manner after knowledge of Applicant's invention calculated to arrive at the rejection. It is clear under the law that this is an impermissible practice. It is not seen how any of the references stand for the propositions advanced by the Examiner in citing sections of the references.

More specifically, the Examiner has cited sections of the references as teaching the specific language of Applicant's claim, but when each section is reviewed, other than for using conventional computer terms like hash tables, i-nodes, etc, do not stand for the propositions advanced. Accordingly, the Examiner is courteously requested to reconsider his rejection in light of the foregoing comments and the amendments to the claim. If this is done, it will become clear that the claimed invention is not anticipated by or obvious from the cited references. Thus, for the foregoing reasons, it is respectfully urged that the claims are clearly in condition for allowance.

Nonetheless, should the Examiner have any comments, questions or suggestions of a nature necessary to expedite the prosecution of the application or to place the case in condition for allowance, he is courteously requested to telephone the undersigned at the number listed below.

Dated: September 21, 2004

Respectfully submitted,



A. José Cortina, Reg. No. 29,733
Daniels Daniels & Verdonik, P.A.
P.O. Drawer 12218
Research Triangle Park, NC 27709
Voice 919.544.5444
Fax 919.544.5920
Email jcortina@d2vlaw.com

Enclosures